AMENDMENTS TO THE SPECIFICATION

Please replace paragraph [0035] with the following paragraph:

[0035] FIG. 2 is a functional block diagram of the ingress point ~~200~~ 130 as a node on, for example, a LAN according to an embodiment of the invention. The ingress point node ~~200~~ 130 may include a first input 210 and a first output 211. The first input 210 and the first output 211 may be indicative of, for example, upstream traffic coming from the WAN 110 to distant LAN users. The ingress point node ~~200~~ 130 may also include a second input 215 coupled with a second output 216. The second input 215 and the second output 216 may indicate, for example, downstream traffic initiated by the LAN users, which may be distant to remote users across the WAN 110. These two streams may be processed by a packet sniffer 230 and raw information, such as, for example, packet delay, jitter, and packet loss may be sent to [[the]] a state machine 220 as input 230. The state machine 220 may receive other inputs as well. For example, input 231 may reflect severity levels calculated at remote ingress points (i.e., at points across the WAN), which are sent to the local state machine 220. Input 233 reflects input to the state machine 220 from network configuration devices. The state machine 220 may include a memory 225 and a processor 224 coupled to the memory 225. The state machine 220 may be configured to control the ingress point call admission policy 240. In one embodiment, the call admission policy may be, for example, a CAC/SM policy 240. Other types of policies, such as, for example, multilevel preemption and precedence policies may be implemented as policies using the methods and systems of the present invention. An output 232 may be produced from the state machine 220 to change CAC/SM policies in 240 ingress point engine, which is collocated with packet sniffer 230. The input 241 to the CAC/SM policy 240 may reflect call/session arrival from the ingress point gatekeepers, which the output 242 from the CAC/SM policy may reflect the admissions, denial, and/or preemption decisions sent to the ingress point gatekeepers. The output 242 from the CAC/SM policy ~~242~~ 240 may be optimized for QoS.

Please replace paragraph [0036] with the following paragraph:

[0036] The processor 224 may be configured to calculate a cost function. Based on the cost function, the processor 224 may determine whether to change a severity level associated with the upstream node. In another embodiment of the invention, the processor 224 may be configured to be coupled to a memory 225, and may implement, based on information stored in memory 225, a CAC/SM policy 240 based on a severity level received via input 233. Based on the CAC/SM policy 240 that may be implemented by processor 224, certain calls may be admitted or blocked [[be]] by the ingress point gatekeepers. In yet another embodiment, processor 224 may be configured to retrieve information associated with a source list and a destination list that may be stored in memory 225, which creates an independent state machine 220 for each of the destination nodes over the WAN 110 that communicate with the local LAN. The processor 224 may also be configured to store information in source and destination lists within memory 225.

Please replace paragraph [0037] with the following paragraph:

[0037] In one embodiment of the invention, the CAC/SM policy 240 may include a multilevel precedence and preemption policy (MLPP), which is capable of scaling network traffic to maintain QoS based on, for example, call/message survivability requirements and/or bandwidth requirements associated with conditions on the network in real-time or near-real-time. The use of an algorithm according to methods of the present invention may result in a network capable of healing congestion due to a sudden surge of traffic demand within, for example, 5 seconds.

Please replace paragraph [0045] with the following paragraph:

[0045] FIG. 5 is a network diagram including two nodes according to an embodiment of the invention. A packet switched core IP network 500 may include a source node 510 and a destination node 530. The ~~Source~~ source node 510 is mathematically connoted by the symbol Ψ, and the destination node is connoted by Y_j. Data may be transmitted from source node 510 to the destination node 530 over IP cloud 520, such as a packet switched IP network. In one example, source node 510 [[Ψ]] has established a call with destination node 530 [[Y_j]] over the IP cloud 520 such that packets of data may flow from source node 510 [[Ψ]] to destination node 530 [[Y_j]]. Source node 510 ~~Node~~ [[Ψ]] may include a memory and a processor associated with the node, as described in more detail with reference to FIG. 2, above. Source node 510 ~~Node Ψ~~ may be configured to add destination node 530 [[Y_j]] to, for example, a destination list. In one embodiment, source node 510 [[Ψ]] may be configured to add all nodes that source node 510 [[Ψ]] communicates with to the destination list. Likewise destination node 530 [[Y_j]] may be associated with a processor and a memory, and may be configured to add source node 510 [[Ψ]] to, for example, a source list.

Please replace paragraph [0047] with the following paragraph:

[0047] Upon receiving a packet, each node may be responsible for determining an end-to-end delay that the packet encountered. The end-to-end delay may be determined based on the difference between the current network time and the time stamped on the packet. In addition to determining the end-to-end delay, the network may be configured to determine a packet loss ratio associated with the transmission of the packets from one end to the other. The packet loss ratio may be calculated based on the packed sequence number that is included on the packets. Thus, source node 510 [[Ψ]] may have information regarding the end-to-end delay and the packet loss for the return trip and destination node 530 [[Y_j]] may have this information for the forward trip.

Please replace paragraph [0048] with the following paragraph:

[0048] The call placed by source node 510 [[Ψ]] to destination node 530 [[Y_j]] may be associated with a particular class of call. Based on the class of the call between source node 510 [[Ψ]] and destination node 530 [[Y_j]], the algorithm may be configured to determine whether the return trip call is violating network QoS requirements. Likewise, destination node 530 [[Y_j]] may be configured to determine if the forward trip call is violating QoS requirements. In one embodiment, source node 510 [[Ψ]] may be configured to update the destination list dedicated for the destination node 530 [[Y_j]] with any change in meeting/violating QoS requirements based on the return trip information. Similarly, destination node 530 [[Y_j]] may be configured to update the entry in its source list associated with source node 510 [[Ψ]] with any change in meeting/violating QoS requirements based on the forward trip information. Destination node 530 ~~Node~~ [[Y_j]] may have the responsibility of reporting to source node 510 [[Ψ]] any significant changes in meeting/violating QoS requirements of the forward trip. This may permit source node 510 [[Ψ]] to have information about the condition both the forward trip and the return trip of the call. In one embodiment of the invention, source node 510 [[Ψ]] may then determine to continue admitting calls/sessions to destination node 530 [[Y_j]] if the severity level does not exceed a predetermined threshold severity level. Alternatively, if the severity level exceeds a predetermined severity level, source node 510 [[Ψ]] may reduce admissions of new calls or sessions to destination node 530 [[Y_j]]. The amount of the reduction by source node 510 [[Ψ]] may depend on the severity level, which in turn may depend on the conditions of all of the calls and sessions. In yet another embodiment, source node 510 [[Ψ]] may be configured to permit the admission of additional calls if the severity level is below a predetermined threshold. In yet another embodiment, the algorithm may be configured to preempt low importance calls to free resources for higher importance calls. In the embodiment illustrated in FIG. 5, source node 510 [[Ψ]] transmits a data stream having an amount of traffic that may be defined by the following equation:

$$T = M_1(t) * P_1 + M_2(t) * P_2 + \ldots + M_n(t) * P_n .$$

Please replace paragraph [0049] with the following paragraph:

[0049] Where $_j(t)$ is the number of calls of a predetermined class of calls over a period of t seconds. P_i is the average message size for a particular class of messages. Based on the traffic received at destination node 530 [[Y_j]], the node may calculate a cost function $S_\Psi(t_k)$. The cost function $S_{105}(t_k)$ may be configured to return a value of -1, 0 or +1 based on, for example, the end-to-end delay, the packet loss ratio, and the last time of update. Destination node 530 ~~Node~~ [[Y_j]] may then update a severity function $V(t_k)$ based on the cost function. Thus, the new severity function looks like:

$$V_\Psi(t_{k+1}) = V(t_k) + S(t_{k+1})$$

Please replace paragraph [0050] with the following paragraph:

[0050] Destination node 530 ~~Node~~ [[Y_j]] may then transmit the severity level $V_\Psi(t_{k+1})$ to source node 510 [[Ψ]] so that source node 510 [[Ψ]] may make any necessary adjustments to the CAC/SM policy. An exemplary listing of severity functions and description of various classes and subclasses of calls appears in Table 1, which follows.

Please replace paragraph [0051] with the following paragraph:

[0051] Assuming that the severity level at t is 3, and that the received traffic T at destination node 530 [[Y_j]] was such that the severity level was increased by 1, source node 510 [[Ψ]] may block all routine VTC and routine voice calls, and all high-resolution VTC calls. Additionally, routine calls may be preempted. The foregoing example and table is meant to be illustrative only and users may define any type of preemption and admission policies that they deem effective depending on the size and type of network as well as the amount of total network traffic.

Please replace paragraph [0056] with the following paragraph:

[0056] With specific reference to forward traffic severity levels, each destination node may keep track of the impact of traffic for each source sending messages to that particular destination node. The severity level can be a summary of the traffic load of the various classes of traffic within the network, and more particularly between a source node and a destination node. In one embodiment, each time the severity level changes, the destination node 530 may send an update severity value to the source node 510.

Please replace paragraph [0057] with the following paragraph:

[0057] In one exemplary embodiment, voice, video and data traffic may travel from a number of source nodes $S = \{\Psi_1, \Psi_2, \Psi_3, \ldots \Psi_n\}$ to a number of destination nodes $D+\{Y_1, Y_2, Y_3, \ldots Y_n\}$. In this exemplary embodiment, Data classes may be designated as $D_1, D_2, \underline{D_3}, \ldots, D_{n1}$, ~~video~~ voice classes may be designated as $V_1, V_2, \underline{V_3}, \ldots, V_{n2}$, and video classes may be designated by VTC_1, VTC_2, $\underline{VTC_3}, \ldots, VTC_{n3}$. These classes may be mapped to $\{C_1, C_2, \underline{C_3}, \ldots, C_n\}$, where C_i corresponds to a class of service and $n = -n1 + n2 + n3$. In this example, for each class of call $\{C_1, C_2, \underline{C_3}, \ldots, C_n\}$ there may be corresponding QoS attributes $\{\underline{Q_1}, Q_2, \underline{Q_3}, \ldots, Q_n\}$, where Q_i $[[=<]] \leq R_i, T_i, E_i, \ldots >$ with R_i is the required fraction of packets of class C_{i1} with end-to-end delay less than T_i seconds, and E_i is an acceptable packet loss ratio.

Please replace paragraph [0058] with the following paragraph:

[0058] In one example, destination node 530 $[[Y_j]]$ may receive a packet of class C.sub.p from source node $[[\Psi_j]]$ 510. When destination node 530 ~~Node~~ $[[Y_j]]$ receives this packet, the algorithm may calculate a cost function and may update the severity level as appropriate, as described above. In one embodiment, if the severity level value changes in value, destination node 530 $[[Y_j]]$ will return the severity value to source node 510 $[[\Psi_j]]$. In an alternative embodiment, the severity value may be returned to source node 510 $[[\Psi_j]]$ regardless of whether it has changed or not. Based on a received severity value from destination node 530 $[[Y_j]]$, source node 510 $[[\Psi_j]]$ may change its admission policy accordingly. Since $S(t_{k+1})$ may depend on the current utilization of the system and is independent of the severity value for the time prior to t_k for any sequence of times $t_0 < t_1 < \ldots < t_{m+1}$, and values $U_0, U_1, \ldots, U_{m+1}$ in the range of the severity values, the severity values may be represented by a Markov chain, as illustrated in FIG. 6. [[the]] The Markov chain may take values in the range of $\{1, 2, \ldots, N\}$. In one embodiment, the cost function S may be defined so that the jitter in the severity levels may be removed. This may be done, for example, by permitting a predetermined time period to pass before S may return a non-zero value. In the Markov chain illustrated in FIG. 6, the source node admission policies (or decisions to change topology) may be made based on the highest of the severity levels. Markov chains allow a probability-based determination based on current conditions to predict what the value of, for example, the next severity value will be.

Please replace paragraph [0068] with the following paragraph:

[0068] After these values are saved, a time period equal to 1 period may pass. The time period may be any acceptable time period, such as, for example, 1 second. This time period may be system dependent. In one embodiment of the invention the time period may decrease if the flow of traffic within the network is high. This may be achieved by keeping a count of the number of packets received within the time period. If the number of packets exceeds a certain threshold, such as, for example, 100 packets before the predetermined time period expires, a new timing may be utilized. For time period t_{k+1}, ~~box~~ step 645, a packet count may be calculated, 655, a packet delay may be calculated 660, and a packet drop ratio may also be calculated 665. In one embodiment, the average jitter may also be calculated. The values from time t_k and t_{k+1} may be compared, step 680, and a value W_{tk+1} may be calculated based on the comparison. The values determined in steps 655, 660, 665, and 675 may be saved, step 677, in a computer-readable medium for comparison purposes, in an embodiment where the algorithm may continue in a loop. If the value of W_{tk+1} is greater than a threshold value (W_{th}), step 685, then the Cost function S, and ergo, the severity value $V(t_{k+1})$ may be updated, step 690. If the value of W_{tk+1} is less than W_{th}, then the cost function may return a value of 0, and the severity function $V(t_{k+1})$ does not need to be updated, step 695. The process may loop back to step ~~645~~ 655 when another packet is received. Another time period equal to, for example, one period may pass and the packet count, packet delay, and the packet drop ratio may be recalculated for the new time period. The algorithm may then determine whether to update the severity level based on these new values as described above.

Please replace paragraph [0069] with the following paragraph:

[0069] FIG. 8 is a flow diagram of a method of determining a CAC/SM policy according to one embodiment of the invention. Initially, a time may be represented by t_{k+n}, where n=0, step 805. The packet may be received at destination node 530 [[Y_j]] at time t_{k+n}, where n=0, step 810. Based on the received packets, the algorithm may calculate a ~~coast~~ cost function $S_j(t_{k+n})$, step 815. In one embodiment, the cost function may return values, such as, for example, -1, 0, and +1. Based on the value returned from the cost function, the severity level $V_j(t_{k+n})$ may be updated, step 825. In one embodiment, when the cost function returns a value of +1, the severity level is increased by 1, when the cost function returns a value of -1, the severity level is decreased by 1, and when the cost function returns a value of 0, the severity level remains unchanged. Once the severity level has been updated, the updated severity level may be returned to the source node [[Ψ]] 510, step 825. In an alternative embodiment, the algorithm may only send the severity level to the source node 510 [[Ψ]] if it has been updated. Once received at the source node [[Ψ]] 510, the algorithm ~~run~~ runs on, for example, the source node processor can determine if the updated severity value ($V_j(t_{k+n})$), which is above a severity value threshold (V_{thS}), step 830. If the severity value is above a threshold, then the algorithm may determine if the severity value exceeds a blocking threshold (V_{thB}) associated with a CAC/SM policy, step 835. If the severity value, $V_j(t_{k+n})$, is above the blocking threshold, V_{thB}, then the algorithm may be configured to block either the lowest class of calls (if no calls are blocked) or the next lowest class of calls (if there are calls blocked), step 840. Time may be allowed to pass to the next period, which is represented by an equivalent block setting n = n+1, where n + 1 represents the next period, step 845.

Please replace paragraph [0071] with the following paragraph:

[0071] In an alternative embodiment, the severity value $V_j(t_{k+n})$ is not compared with a predetermined threshold, but rather may be referenced against, for example, a table, such as a look up table to determine a CAC/SM policy that reflects current network conditions. For example, the severity level received from destination node 530 [[Y_j]] may be compared to a table associated with the CAC/SM to determine if any calls should be blocked or admitted. In another alternative embodiment, entire classes of calls are not blocked, [[by]] but calls of a particular class having a size larger than a threshold size may be blocked. Therefore, the CAC/SM policy may improve or maintain QoS by limiting the amount of total traffic on the network by simply reducing the size of the calls and sessions admitted over the network.

Please replace paragraph [0072] with the following paragraph:

[0072] FIG. 9 illustrates a Packet Switched IP Network topology according to another embodiment of the invention. A multiple node network is illustrated in FIG. 9, including a number of source nodes 910 (Y_i), which transmit data to node 930 (v), and a number of destination nodes 920 (X_i), which receive data from node 930 over packet switched IP network 940.

Please replace paragraph [0073] with the following paragraph:

[0073] For each of the destination nodes 920, the destination list established by the algorithm may include a destination node identifier (ID) which may specify the unique identification of the node on the network; the forward traffic severity level which is reported by the destination node 920, as described above; the last update time, which indicates the last time the severity level was updated; the total number of received packets of classes C_1-C_n (broken down by class), the total number of packets violating end-to-end delay times and the packet loss associated with each class of packets sent. Additionally, the destination list may also include the return traffic severity level, which indicates the severity level associated with the return traffic for each of the destination nodes 920. Similarly, for each of the source nodes 910, the source list may contain a source node identifier (ID), a forward traffic severity level, and the last update time, which indicates the last time the severity level was updated; the total number of received packets of classes C_1-C_n (broken down by class), the total number of packets violating end-to-end delay times and the packet loss associated with each class of packets received.

Please replace paragraph [0074] with the following paragraph:

[0074] In one embodiment, the algorithm may be configure to drop old information by factoring down the collected counters every period. This may ensure that the counters have more recent information ~~base don~~ based on the recent network conditions. Additionally, in some embodiments, the algorithm may continuously look at the collected information for each data structure, ~~compares~~ compare it to the values of QoS attributes for each class of service (% packets to meet certain end-to-end delay, % of allowed packet loss), and may generate a severity level for each source node 910 ("X_1", "X_2", "X_3", "X_4" and "X_5"). Any change in the severity level may be reported to the source node using, for example, a token.

Please replace paragraph [0075] with the following paragraph:

[0075] In yet another embodiment of the invention, the algorithm may include a method that is capable of distinguishing between the loss of a signal or a signal blockage and a sudden arrival or a surge in network usage, and therefore, may apply a different CAC/SM policy depending on the particular deleterious effects on the network.

Please replace paragraph [0076] with the following paragraph:

[0076] FIG. 10 is a flow diagram of a method of determining when network behavior is due to sudden arrival or blockage according to an embodiment of the invention using, for example, a window of values. After a packet is received at a node, the packet loss may be determined, step 1010. Once the packet loss is determined, the packet loss determined in step 1010 may be compared to a packet loss tolerance, step 1015. In one embodiment, the packet loss tolerance may be, for example, a 10% loss. Any packet loss tolerance may be used in connection with the present invention according to the particular characteristics of the network. If the packet loss tolerance is not violated, step 1020, then the algorithm will wait until the next time period, step 1080 and then repeat the loop. If the packet loss tolerance is violated, the end-to-end delay may be determined, step 1025. The delay may be compared to a delay threshold, step 1030. Then the jitter is determined, step 1040. The jitter is compared to a jitter threshold 1050. A determination may be made to see if both the jitter and the delay violate their respective thresholds, step 1055. If both jitter and delay exceed their thresholds, then the violation in packet loss may be due to sudden arrival, step 1060, and the algorithm may take large steps up the Markov chain 1065. The algorithm may permit conditions to remain unchanged until the next time period, step 1080, when the algorithm may repeat the loop. If jitter and delay are not above the thresholds, then the increase in packet loss may be due to a blockage, step 1070, and the algorithm may be configured to take small steps up the Markov chain, step 1075. The algorithm may permit conditions to remain unchanged until the next time period, step 1080, when the algorithm may repeat the loop.